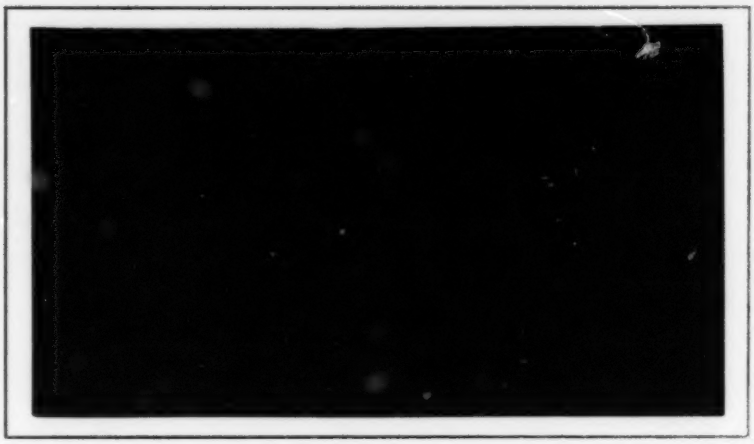
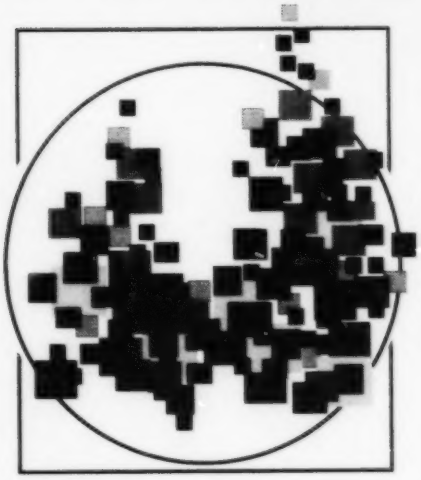




CENTRE FOR
SYSTEMS
SCIENCE

Simon Fraser University, Burnaby, B.C., Canada V5A 1S6 / (604) 291-4704

92-05396/

On Linear Generalized Hough Transform And Its Parallelization

Z. N. Li, B. Yao and F. Tong

CSS-IS TR 91-03

May 1991

ON LINEAR GENERALIZED HOUGH TRANSFORM AND ITS PARALLELIZATION

Z.N. Li, B. Yao, and F. Tong

School of Computing Science
Simon Fraser University
Burnaby, B.C. V5A 1S6
Canada

ABSTRACT

The Generalized Hough Transform (GHT) proposed by Ballard is an effective method for recognizing objects of arbitrary shape. It converts a problem of global pattern matching into local peak searching. However, the GHT has difficulty handling occluded objects. It also suffers from the congestion at the single peak point when attempting its potential parallelization. In this paper, we will present a Linear Generalized Hough Transform (LIGHT) in which a linear numeric pattern is used to replace the single peak in the GHT. The LIGHT is more capable of detecting partially occluded objects. Moreover, it is well-suited for parallelization, especially on SIMD array processors. Several sequential and parallel LIGHT algorithms and our preliminary results will be presented. The result for parallel implementation was obtained from an SIMD one-dimensional array processor (the AIS-4000 Vision Processor with 512 processing elements). A speed-up factor of two orders of magnitude was achieved.

Key Words: Linear Generalized Hough Transform, Object Recognition, Parallel Processing

1. INTRODUCTION

The Generalized Hough Transform (GHT) was proposed by Ballard in 1981 [3]. It is an effective method for recognizing objects of arbitrary shape that cannot readily be described in analytic forms. It converts a problem of global pattern matching into local peak searching.

The innovative ability of object recognition provided by the GHT has attracted many researchers. Since 1981, the field of computer vision and pattern recognition has seen many successful GHT applications under various environments. For example, Adiv applied the GHT to match patterns in two different images resulting from motion [1]. He proposed to use coarse parameter resolution for large objects, and to partition the image into small sub-images for detecting small objects in order to reduce the computational cost and also to cope with the problem that small objects only produce small peaks in parameter space. Kasif et al. [11] applied the GHT to match subgraphs derived from geographical maps. They compared the GHT with other methods that could be used for solving the subgraph isomorphism problem and pointed out the possibility of implementing their method on a parallel machine. In their attempt to extend the GHT to 3D object recognition, Ballard and Sabbath [4] used the Hough technique to detect groups of parameters sequentially and showed how to derive constraints within the groups. Davies has published many papers on his studies of the GHT. He has used the GHT to detect both sharp and blunt corners [8]. He showed that the choice of reference point could have impact on both the accuracy of corner location and the sensitivity for corner detection. He discussed the best compromise of the two factors. Illingworth and Kittler have published an extensive survey on various researches on the Hough transforms [10].

There have been some studies on extending Ballard's work to detect partially occluded objects. Davis [7] suggested a hierarchical approach of the GHT. His method was to decompose the complex shape into simpler shape. The GHT was then applied on each part, requiring the detection of smaller parts before detecting the composite. The shape-recognition process was better controlled and carried out using a top down hierarchical approach. This made the detection of a partial shape possible. However, Davis' method required the difficult model decomposition problem to be solved a priori.

In their survey, Illingworth and Kittler [10] showed many attempts on parallel implementations of the Hough transform. Most of them were for 2D line and curve detection [6, 12, 14]. The Hough space was distributed among participating Processing Elements, or PE's. The studies had been focused on determining the best communication routing among PE's that yielded the least contention. Little was known about applying parallel processing to the GHT.

One of the most commonly used architectures for parallel image processing and analysis is the Single-Instruction Multiple-Data (SIMD) array. They are of the forms of mesh-array or 1D linear array which are both commercially available. Usually, each PE is connected to its neighbors and has its own local memory. Most existing array processors do not support indirect addressing. As a result, all of the PE's can only access their local memory using a same offset at a time (e.g., MPP [5], AIS [16]). The new generation array processors are likely to provide the indirect addressing mode for PE's to access their local memory locations with different offsets simultaneously (e.g., the proposed CENTIPEDE machine [17]).

In this paper, a Linear Generalized Hough Transform (LIGHT) is presented in which a linear numeric pattern is used to replace the single peak in the GHT. The LIGHT is more capable of detecting partially occluded objects. Moreover, it is well-suited for parallelization, especially on the SIMD array processors. We will present our sequential Planar-LIGHT (P-LIGHT) and Spatial-LIGHT (S-LIGHT) algorithms and our parallel implementation of the S-LIGHT algorithm. Preliminary results will be shown to illustrate the recognition of partially occluded objects from both real-images and synthetic images. The result for parallel implementation was obtained from an SIMD one-dimensional array processor (the AIS-4000 Vision Processor with 512 processing elements). A speed-up factor of two orders of magnitude was achieved.

Section 2 briefly describes the GHT algorithm and its limitations. Section 3 introduces our LIGHT method and the P-LIGHT and S-LIGHT algorithms. The parallel implementations for the S-LIGHT algorithm in linear array processors with or without the indirect addressing mode are presented in Section 4. Section 5 presents some implementation details and experimental results from both Sun workstations and our AIS-4000 parallel vision processor. Section 6 concludes the paper.

2. THE GENERALIZED HOUGH TRANSFORM AND ITS LIMITATIONS

In 1981, Ballard extended Merlin and Farbers' work [13] and proposed the Generalized Hough Transform (GHT) for detecting arbitrary shape of any orientation and scale [3]. Suppose that the shape of a 2D object cannot be described by analytical equations. In Ballard's GHT, a model image is used to construct a so-called R-table. A polar coordinate system is established in the model image by fixing a reference point and using it as the origin. The R-table records the positions of all boundary points by their polar coordinate values. The rows of the R-table are indexed by the gradient directions of the edge points on the boundary.

During the object recognition process, a 2D array can be used as an accumulator, which is called Hough space or parameter space. Edge points will cast votes to the hypothetical reference point in the Hough space according to their gradient directions and the corresponding R-table entries. If the object in

the image is identical to the model, then the cell in the accumulator array that corresponds to the reference point will obtain the highest number of votes. The peak value would be equal to the number of boundary points of the object when the model and the object match perfectly. In case the scale and/or orientation of the object in the image are different from the model, various scales and orientations must be examined.

The GHT can be viewed as an evidence-gathering process. When the boundary of the object is not perfect (e.g. due to image noises, object deformation, or occlusion), the peak value will drop by a certain percentage. As Grimson and Huttenlocher [9] pointed out that, while the GHT could tolerate slightly deformed shapes, it might also give false results since it was difficult to know whether a peak was from a slightly deformed shape or from a similar shape. Moreover, the GHT cannot effectively recognize partially occluded objects, since the only information available is the peak value. Although Davis [7] proposed a method to solve the occlusion problem by decomposing the complex model into simpler subtemplates, he did not suggest how to achieve a good decomposition of the model so that the GHT can be used effectively to recognize the non-occluded parts.

The GHT has its potential for parallel processing since each edge point votes independently. However, the reference point that accumulates the votes from all the edge points is a congested spot which causes contentions among the PE's, and there is no obvious way to solve this problem.

3. A LINEAR GENERALIZED HOUGH TRANSFORM

The essence of the LInear Generalized Hough Transform (LIGHT) is to use a reference axis to replace the single reference point used in the GHT. A model is described by a linear pattern, and during the process of object recognition there will be linear patterns accumulated in the Hough space. An object is detected when there is a match between a linear pattern and the model description.

3.1 Planar Linear GHT Algorithm (P-LIGHT) for Object Recognition

In this section, a *Planar LInear Generalized Hough Transform (P-LIGHT)* will be introduced to demonstrate how the LIGHT algorithm can be used to detect some (possibly partially-occluded) objects. The complexity of the algorithm and its limitations will then be discussed.

3.1.1 Model Construction

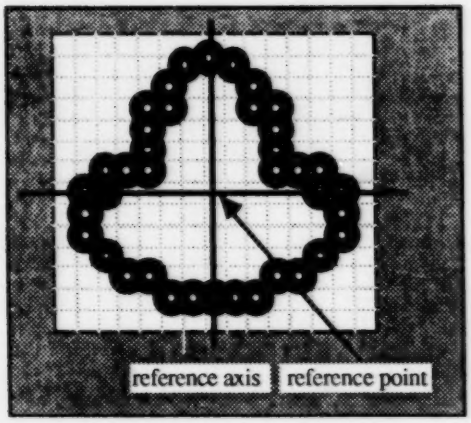
A model which constitutes a VX-table and a vertical pattern (V-pattern) is constructed for each object that is to be recognized. First, a *reference axis* is set on the given object. Without loss of generality, a vertical reference axis which cuts through the model object in the image will be chosen. Unless specified otherwise,

the Y-axis and the reference axis will be used interchangeably in the following discussion. An arbitrary point (e.g. the origin) on the reference axis can be chosen as a *reference point.*

A *VX-table* is constructed for each model. As in the original GHT, the gradient direction ϕ for the boundary points of the object is used to index the VX-table. Each entry in a specific row of the VX-table records a position of a boundary point which has the gradient direction ϕ. The position of a boundary point is represented by the distance from the point to the Y-axis, i.e., its X-coordinate value. The vertical distance from the point to the reference point, i.e., its Y-coordinate value, is not recorded for the P-LIGHT algorithm. If more than two edge points from a same image column (with a same X-coordinate value) have an identical gradient direction ϕ, then they will share a same entry in the VX-table. Hence, the total number of entries in a VX-table may be less than the number of object boundary points. Consequently, the width m of the VX-table may be smaller than the width of the R-table in the GHT.

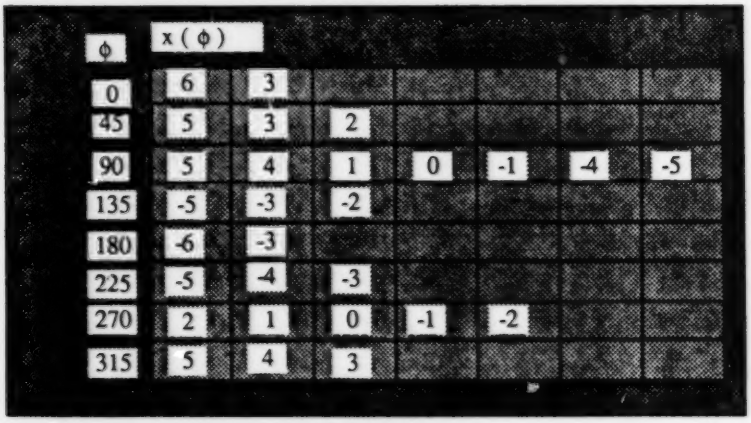
The linear pattern as shown in Figure 3.1 is called *Vertical pattern (V-pattern).* It has the following properties: (a) The length of the V-pattern is equal to the length of the object along the direction of the reference axis, and (b) The number of boundary points that can be projected onto a single point on the reference axis is the value that is stored in the corresponding entry of the V-pattern. In short, the V-pattern reflects subtotals of object boundary points on each image row.

Figure 3.1 shows an example of an object, its V-pattern, and its VX-table. Some simple gradient directions (e.g. 0, 45, 90 degrees, etc.) are assumed.

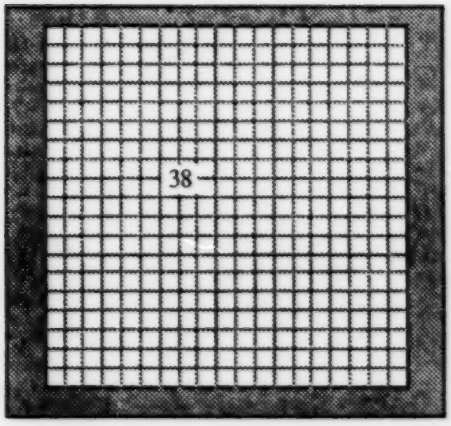


(a)

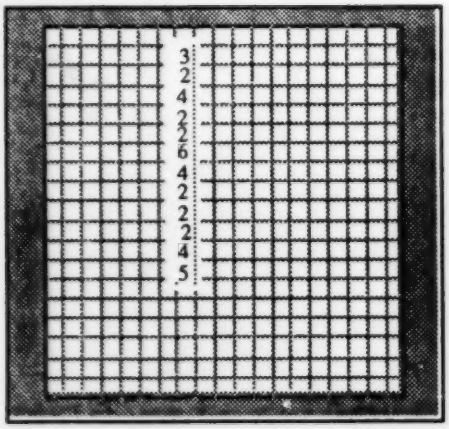
(b)



(c) VX-table



(d) Hough space accumulated by GHT



(e) Hough space accumulated by P-LIGHT

Figure 3.1 An Example of Model Construction and Vote Accumulation

3.1.2 The P-LIGHT Algorithm

The P-LIGHT algorithm for locating an object is described in this section. The handling of scale and rotation of objects is similar to what is done in the GHT and hence is not described here.

P-LIGHT Algorithm:

(1) Initialize a 2-D Hough space (accumulator A), i.e.,

$A(x, y) = 0$ for all (x, y).

(2) (Vote Accumulation)

For each edge point (x, y) of gradient direction ϕ do

For each VX-table entry $x(\phi)$, increment the accumulator array, i.e.,

$$x_c = x - x(\phi),$$

$$y_c = y,$$

$$A(x_c, y_c) = A(x_c, y_c) + 1 . \qquad (3.1)$$

(3) (Pattern Matching)

For any i, j, if $A(x_i, y_j)$, $A(x_i, y_{j+1})$, ..., $A(x_i\ y_{j+L-1})$ has a complete (or partial) match with the V-pattern, then the object sought exists (partially) in the input image. Column x_i indicates the location of the reference axis for the (partial) object and the matched segments correspond to the visible parts of the object.

(4) Stop.

For the comparison of the respective accumulation results by the GHT and the P-LIGHT algorithms, an example is illustrated in Figure 3.1(d) and 3.1(e). To clearly indicate the difference between our technique and the GHT, only the contents in the Hough spaces that are of interest are shown.

3.1.3 Discussion On Pattern Matching

The LIGHT algorithm requires a complete or partial matching between the V-pattern and the patterns accumulated in the columns of the Hough space as shown in Step 3 of the P-LIGHT algorithm. This may not be very difficult for ideal images where there are no distortions and every object boundary is in perfect shape. Some special cases of partial occlusions may be tolerated when segments of the object boundary are either intact or completely occluded. For such images the final pattern match becomes a problem of finding the longest common substrings between the V-pattern and the patterns in the Hough space. The longest

common substring algorithm has been shown in [2] and the time complexity is linear to the length L of the string.[1]

Practical computer vision applications often involve images that contain deformed object boundaries and noises. Therefore, the vote accumulation in the Hough space will not yield an intact linear pattern or even a good segment of the pattern, even if the object is in the image. A systematic study of heuristic tolerance criteria for inexact matching is needed for these more difficult situations. However, such study is beyond the scope of this paper and will be a part of the future research. All the discussions in this paper will assume the above mentioned ideal images with possible occlusions. Nevertheless, some good empirical heuristic measures are derived for inexact pattern matching in recognizing objects from real-world images. The experimental results will be shown in Section 5.

3.1.4 Complexity Analysis for the P-LIGHT Algorithm

The Hough space used in the P-LIGHT algorithm is simply a two dimensional N×N array based on the image size of N×N. Hence, the P-LIGHT algorithm has $O(N^2)$ space complexity which is the same as the GHT.

The vote accumulation in the P-LIGHT algorithm usually takes less time than in the GHT, because the VX-table used by the P-LIGHT algorithm may contain less entries than the R-table used by the GHT algorithm. However, in the worst case, both tables are of the same width m. The worse case time requirement for the vote accumulation is $O(N^2m)$, where m is the maximum number of columns in the VX-table. For the type of simple occluded objects that the P-LIGHT can deal with, efficient numeric string matching algorithms presented in [2] can be employed for a complete V-pattern search in Hough space, and the time complexity for the pattern matching is $O(N^2)$ in a N×N Hough space. Overall, the time complexity of the P-LIGHT would be at the same order as the GHT algorithm.

3.1.5 Limitation of The P-LIGHT Algorithm

The P-LIGHT algorithm is relatively simple and efficient. It is useful for simple cases, e.g., only a single object in the image. It is not robust in dealing with partially occluded objects. The following example shows the limitation of the P-LIGHT algorithm. Assume the object has a calabash-shape as shown in Figure 3.1(a), Figure 3.2(a) is an input image which contains two of such overlapped objects (their reference axes are also overlapped). If the P-LIGHT algorithm is applied to the image, the resulting pattern

---

[1] Throughout this paper, the worst-case complexity will be used.

is a partial overlapping of the two V-patterns (shown in Figure 3.2(c)). The P-LIGHT algorithm will fail to recognize the overlapped V-pattern.

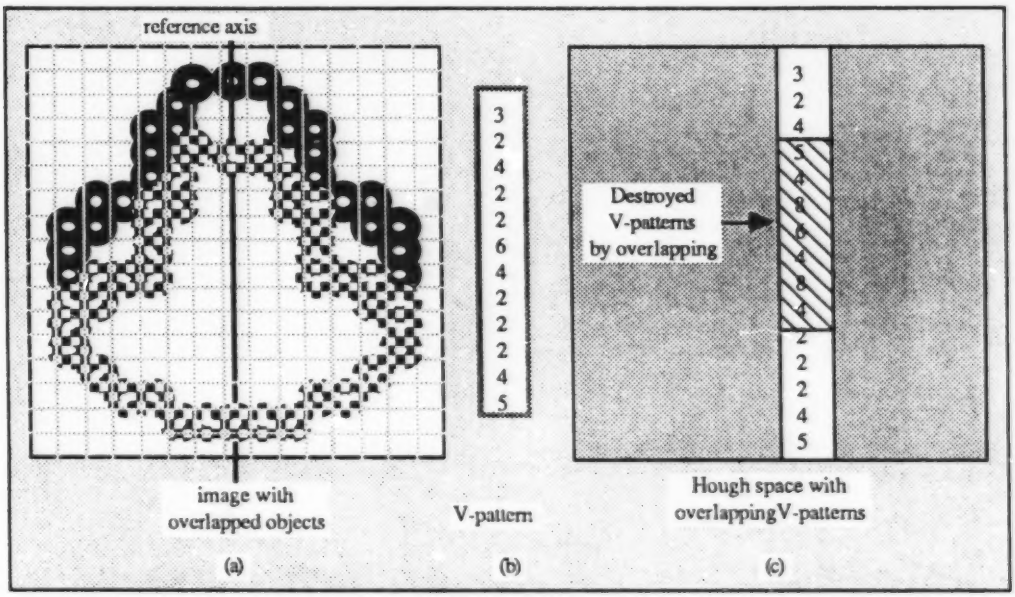

reference axis
image with overlapped objects
(a)
3 2 4 2 2 6 4 2 2 2 4 5
V-pattern
(b)
Destroyed V-patterns by overlapping
3 2 4 5 4 8 6 4 8 4 2 2 2 4 5
Hough space with overlapping V-patterns
(c)

Figure 3.2 Destroyed V-patterns by Overlapping

One might argue that the overlapping of reference axes for the two objects in Figure 3.2(a) can be avoided by choosing a different reference axis that is not along the vertical direction for this pair of calabash-shaped objects. However, when more complicated cases arise, the P-LIGHT algorithm may fail again.

3.2 Spatial Linear GHT Algorithm (S-LIGHT)

The above major limitation of the P-LIGHT algorithm is caused by the overlapping V-patterns in the two-dimensional Hough space. In fact, it is inherent from our extension to the GHT algorithm in which a point peak is extended into a linear pattern in the Hough space. The solution to this is to expand our two-dimensional Hough space into a three-dimensional (spatial) Hough space. To achieve this, we will employ both the reference point used in the GHT and the linear pattern technique used in the P-LIGHT. The unique reference point for each object is used as the distinct stand for the accumulation of V-pattern for that object in the 3D Hough space. Figure 3.3 illustrates this idea. Intuitively, the V-pattern can be considered as "standing" on the reference point. The size of the 3D Hough space is N×N×L, where L is the length of the V-pattern. By such expansion of the Hough space, the overlapping problem shown in Figure 3.2 is solved,

since the V-patterns for the two overlapped objects will now "stand" on two distinct points, each corresponding to the reference point of one object. The new V-pattern accumulation for the two objects shown in Figure 3.2(a) is shown in Figure 3.4.

The improved algorithm is called the *Spatial LInear Generalized Hough Transform (S-LIGHT)* algorithm.

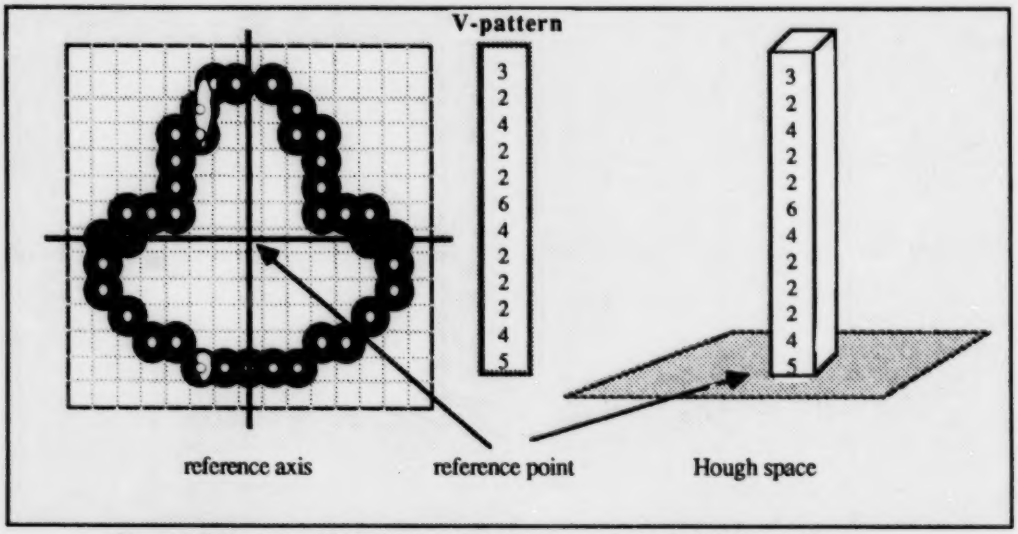


Figure 3.3 V-pattern "Standing" On The Reference Point In 3D Hough Space

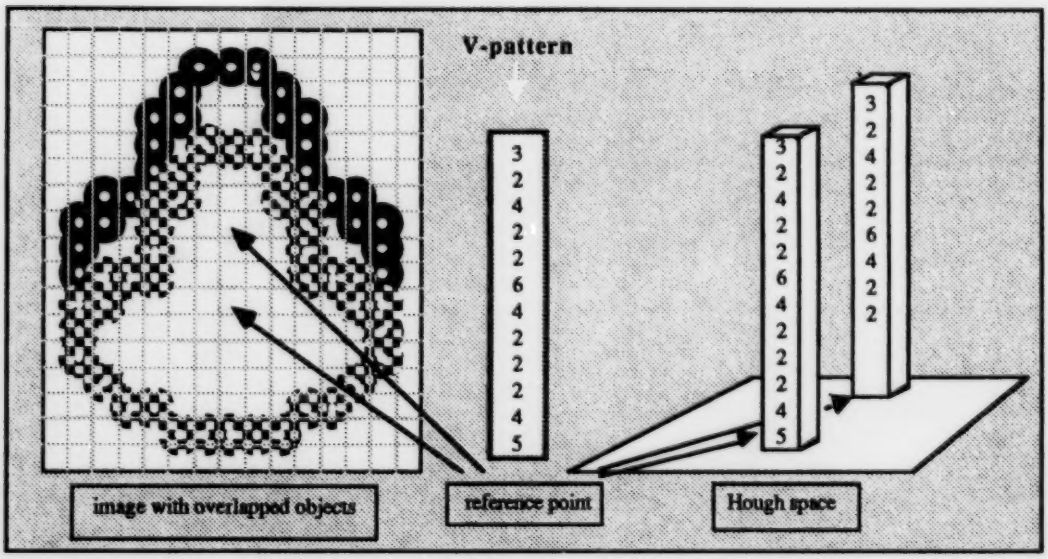


Figure 3.4 Non-overlapping Accumulation of V-patterns

3.2.1 A New Table for the S-LIGHT Algorithm

As each boundary point must now vote to a V-pattern standing on a reference point, the VX-table has to be expanded so as to include the y coordinate information as well. The expanded VX-table is called *VXY-table*. The number of its entries is now exactly the same as the number of boundary points on the model. The VXY-table is actually the same as one of the possible forms of the R-table for the GHT. Figure 3.5 shows the VXY-table for the model in Figure 3.1(a).

ϕ	(x (ϕ) , y (ϕ))						
0	(6, 0)	(6,-1)	(6,-2)	(3, 2)	(3, 3)	(3, 4)	
45	(5, 0)	(3, 1)	(2, 4)	(2, 5)			
90	(5, 1)	(4, 1)	(1, 6)	(0, 6)	(-1, 6)	(-4, 1)	(-5, 1)
135	(-5, 0)	(-3, 1)	(-2, 4)	(-2, 5)			
180	(-6, 0)	(-6,-1)	(-6,-2)	(-3, 2)	(-3, 3)	(-3, 4)	
225	(-5,-3)	(-4,-4)	(-3,-4)				
270	(2,-5)	(1,-5)	(0,-5)	(-1,-5)	(-2,-5)		
315	(5,-3)	(4,-4)	(3,-4)				

Figure 3.5 VXY-table built for the object in Figure 3.1(a)

3.2.2 The S-LIGHT Algorithm

The S-LIGHT algorithm is more robust in detecting partially occluded objects and is presented below:

S-LIGHT Algorithm:

(1) Initialize the 3-D Hough space (accumulator A) of size N×N×L, i.e.,

$A(x, y, z) = 0$ for all (x, y, z).

(2) (Vote Accumulation)

For each edge point (x,y) of gradient direction ϕ do

For each VXY-table entry ($x(\phi)$, $y(\phi)$), increment the accumulator array, i.e.,

$$x_c = x - x(\phi),$$

$$y_c = y - y(\phi),$$

$$z_c = y(\phi),$$

$$A(x_c, y_c, z_c) = A(x_c\ y_c\ z_c) + 1\ . \qquad (3.2)$$

(3) (Pattern Matching)

For any i, j, if $A(x_i, y_i, 0)$, $A(x_i, y_i, 1)$, ..., $A(x_i, y_i, L-1)$ has a complete (or partial) match with the V-pattern, then the object sought exists (partially) in the input image. The (x_i, y_i)

indicates the location of the reference point of the (partial) object and the matched segments in the standing column correspond to the visible parts of the object.

(4) Stop

3.2.3 Complexity Analysis for the S-LIGHT Algorithm

Assume that the size of the input image is N×N, the width of the VXY-table is m, and the length of the V-pattern is L:

- The space requirement for the 3D Hough space is $O(N^2L)$.
- Since there are at most $O(N^2)$ edge points voting and each of them refers to at most O(m) entries in the VXY-table, the worst-case running time for the vote accumulation is $O(N^2m)$.
- It is important to point out, that in the pattern matching step for the S-LIGHT algorithm, there is no need to examine any entries of the Hough space more than once. Since $z_c = y(\phi)$, the vertical offset (height) of each entry of the V-pattern in the 3D Hough space is predetermined by $y(\phi)$. Whether it is an exact matching algorithm for ideal images or inexact matching for other images, each entry in the Hough space will be examined exactly once. Therefore, Step 3 in the S-LIGHT algorithm has an $O(N^2L)$ time complexity.

Overall, the time complexity of the S-LIGHT algorithm is $O(N^2(m+L))$. The time complexity for the vote accumulation in the GHT is also $O(N^2m)$. The S-LIGHT algorithm spends more time in pattern matching in the 3D Hough space while the GHT merely searches a 2D Hough space for peaks.

The space complexity for the S-LIGHT algorithm is $O(N^2L)$, which is a factor of L higher than the GHT or P-LIGHT. In Section 5 a slicing method will be introduced to reduce the space requirement for actual implementation.

4. PARALLEL ALGORITHMS FOR S-LIGHT

The S-LIGHT algorithm was shown to be more powerful in detecting partially occluded objects than the GHT and P-LIGHT algorithms. Unfortunately, it has a time complexity of $O(N^2(m+L))$ which in the worst case is $O(N^3)$, when the length of the object is in the order of the image size (L=N). In this section, we will develop and analyze parallel algorithms based on the S-LIGHT scheme that can be implemented on a one-dimensional *Linear Array Processors (LAP)*. The LAP is one of the simplest forms of the SIMD mesh array architectures [15], and is commercially available [16].

4.1 Configuration of the 3D Hough Space

The configuration of the 3D Hough space is illustrated by Figure. 4.1. Given a one-dimensional SIMD array processor with N PE's which are linked into a ring and an input image frame of size N×N, we adopt a convenient way of assigning PE's to the image, i.e., each PE is assigned to a row in the image frame.

The N×N×L 3D Hough space is distributed to N PE's. A 2D array of L rows by N columns in each PE's local memory s allocated. We call such a 2D array *Distributed Hough Space (DHS)*. Each column of a DHS corresponds to an image point.

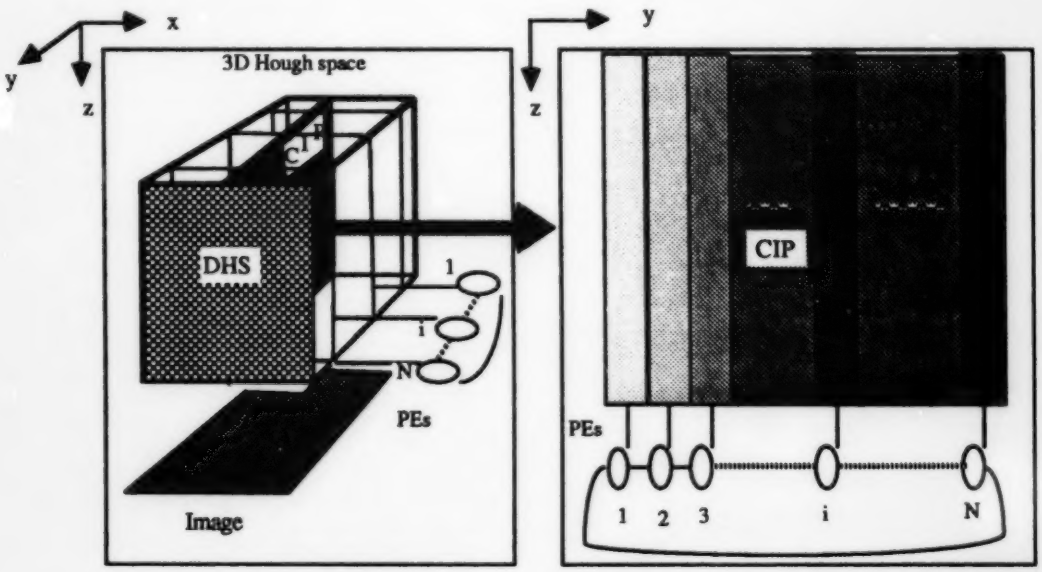


Figure 4.1 Configuration of the 3D Hough Space

Since there is a total of N PE's, the total number of DHS's is N and each of the DHS's can be numbered according to the identification number of the PE that contains the DHS in its local memory. By lining up all the N DHS's according to their sequential numbers, a N×N×L 3D array is formed which is equivalent to the 3D Hough space used in the S-LIGHT algorithm.

A *Cross Intersect Plane (CIP)* in the 3D Hough space is defined as a plane containing N columns, one column from each of the N DHS's. Each CIP corresponds to one column in the image. A CIP is also illustrated in Figure 4.1.

4.2 Diagonal V-Pattern Accumulation

During the process of vote accumulation using a LAP, the PE's will sweep across the image plane, processing one CIP at a time. There are potentially large amount of communications among PE's. The PE's whose image rows contain boundary points will vote and communicate with the PE that contains the reference point, since the accumulated V-pattern will be stored in the local memory of this latter PE. This creates a congested spot. Such communications could usually be fulfilled through "shift" operations in the LAP which require as many as O(N) steps. For example, suppose there is an edge point at the y-th row and the x-th column. Its corresponding VXY-table entry has offset values (a,b). According to the Equation 3.2, the voting destination is (x-a, y-b, b) in the 3D Hough space. Because the y-th PE is in charge of the image point (x, y) and the voting destination (x-a, y-b, b) is in the DHS of the (y-b)-th PE's local memory, the vote must be shifted by b (or (N-b)) times to reach its destination. This will generate the entire V-pattern in a single column of the CIP as shown in Figure 4.2(a).

The above shift operation had to be repeated by the LAP on N CIP's, and for each one of the O(md) VXY-table entries, where d is the number of rows and m is the maximum number of columns in the VXY-table. Hence, the shift could be an expensive $O(N^2md)$ operation on the LAP. To reduce the communication cost among PE's, we propose that each PE defines a voting destination in its own local memory. The V-pattern will be accumulated diagonally in a single CIP with its bottom on the reference point as shown in Figure 4.2(b). This will reduce the cost of the voting for each VXY-table entry on each CIP to a constant and the total voting cost to O(Nmd). This kind of *diagonal V-pattern accumulation* is made possible by the fact that there is a one-to-one mapping between the individual V-pattern entries and the horizontal planes in the 3D Hough space.

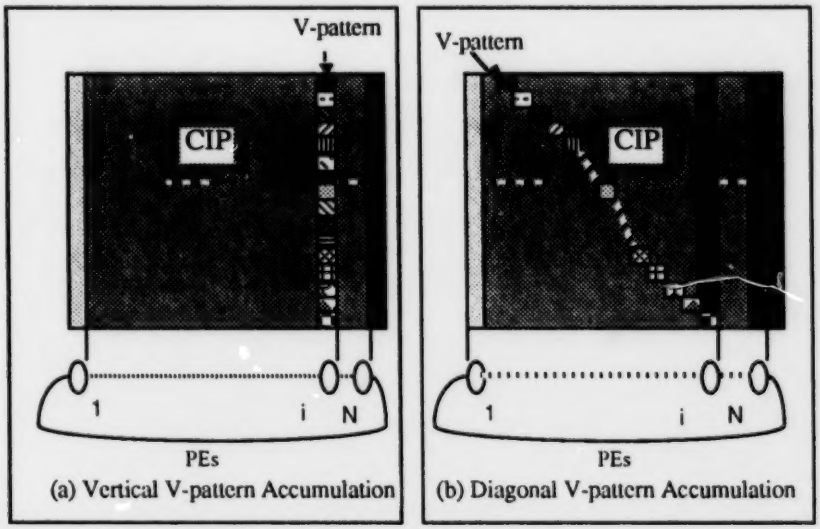


Figure 4.2 Diagonal V-pattern Compared with the Vertical One

4.3 Algorithm for Array Processors without Indirect Addressing

In this section, we will present our first *Array processor S-LIGHT (AS-LIGHT)* algorithm which utilizes the diagonal V-pattern technique. The parallel machine model used in this section does not require PE's to perform indirect addressing. The central controller will look up the VXY-table entries one at a time. The PE's will access their own local memory and use the same offset to perform voting. This requires a very basic SIMD one dimensional linear array processor. Our AIS parallel vision processor is basically such a LAP. In Section 5, we will show some preliminary result for AS-LIGHT obtained from the AIS machine.

4.3.1 The AS-LIGHT Algorithm

AS-LIGHT Algorithm:

(1) Each PE initializes its own DHS at the same time.

(2) (Vote Accumulation)

For x = 1 to N do

For j = 1 to d do /* d - # of rows in VXY-table. */

For i = 1 to m do /* m - # of columns in each row in VXY-table. */

Each PE_y (y = 1, ..., N) performs the following voting procedure at the same time:

if the x-th pixel is an edge point of gradient direction ϕ_j and the contents $(x(\phi_j)_i$, $y(\phi_j)_i)$ of VXY-table$(\phi_j)_i$ is received from the central controller, then

$x_c = x - x(\phi_j)_i$,

$z_C = y(\phi j)i$,

$A(x_C, y, z_C) = A(x_C, y, z_C) + 1$. (4.1)

endif

(3) (Pattern Matching)

For x = 1 to N do: Call procedure "Diagonal V-pattern Search" (explained below)

(4) Stop.

Note that the voting by Equation 4.1 in Step 2 takes place in each PE's local memory, i.e., there are no communications between any PE's. The contents of the VXY-table are looked up and sent by the central controller to all the PE's, one VXY-table entry at a time, and each PE votes to one of its local memory addresses at a time based on the VXY-table entry received.

In Step 3 of the above algorithm, search for the V-pattern is carried out in the diagonal direction in each CIP as shown by Figure 4.2. The matching information is pieced together by the PE's along the diagonal direction in a CIP. Each PE computes a partial solution which can be symbolically expressed as, e.g., "matched from 1 to 1". It is then passed on to its succeeding neighboring PE. During the next iteration, the PE receives information from its preceding neighboring PE, and will carry on the comparison of its present entry in Hough space with the corresponding V-pattern entry. The results are merged to get a new expanded partial solution, e.g., "matched from 1 to 2", and so on. This process continues until the whole V-pattern has been compared once. The search then continues on the next CIP. There is a total of N CIP's to be searched.

To give a concrete example of the diagonal V-pattern searching, we associate a binary matrix B of size N×L to each CIP (also size N×L). We will focus our following discussion on the search in one CIP, since the search of the other N-1 CIP's is done in the same way. The entry (i, j) in the binary matrix is set to 1 if, and only if, the entry (i, j) in the CIP matches the j-th entry in the V-pattern. An example of the binary matrix is shown in Figure 4.3(a). This binary matrix can be constructed in O(L) steps by N PE's.

The procedure to search the V-pattern is described as follows, "first" and "last" are PE's' local variables to record the starting and ending points of the matched segments.

Procedure Diagonal V-pattern Search (in a single CIP)

Set all PE's (first, last) = (nil, nil).

For z = 1 to L do (all PE_y, y = 1, 2, ..., N do at the same time)

If B(y, z) = 1 then if first ≠ nil then PE_y shift (first, last+1) to $PE_{y+1 \bmod N}$;

else PE_y shift (z, z) to $PE_{y+1 \bmod N}$;

else if first = nil then PE_y shift (nil, nil) to $PE_{y+1 \bmod N}$;

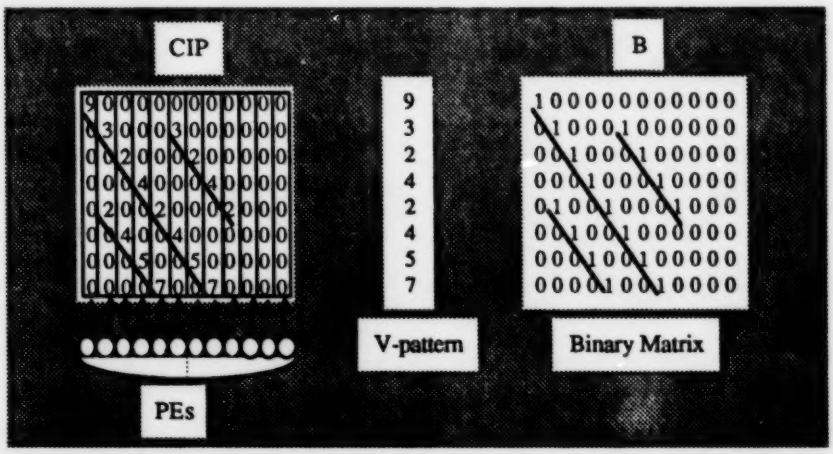
else {declare (first, last) a matched segment;
set (first, last) = (nil, nil); }

PE_y receives (first, last) from PE_{y-1} mod N.

EndFor-loop

EndProcedure

An example of the search for diagonal V-pattern matching is shown in Figure 4.3(b).



(a)

step	Shifting (first,last) among PEs											
	1-->2	2-->3	3-->4	4-->5	5-->6	6-->7	7-->8	8-->9	9-->10	10->11	11-->12	12-->1
1	(1, 1)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
2	(nil,nil)	(1, 2)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 2)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
3	(nil,nil)	(nil,nil)	(1, 3)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 3)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
4	(nil,nil)	(nil,nil)	(nil,nil)	(1, 4)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 4)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
5	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 5)	(nil,nil)	(nil,nil)	(nil,nil)	(2, 5)	(nil,nil)	(nil,nil)	(nil,nil)
6	(nil,nil)	(6, 6)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 6)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
7	(nil,nil)	(nil,nil)	(6, 7)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 7)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
8	(nil,nil)	(nil,nil)	(nil,nil)	(6, 8)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 8)	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)
9	(nil,nil)	(nil,nil)	(nil,nil)	(nil,nil)	(6, 9)	(nil,nil)	(nil,nil)	(nil,nil)	(1, 9)	(nil,nil)	(nil,nil)	(nil,nil)

(b)

Figure 4.3 An Example of the Search for Diagonal V-Pattern Matching

The two most important properties of the AS-LIGHT algorithm are:

(1) There were no write conflicts among PE's. During the voting process, each PE only works in its local memory without communicating with others.

(2) After the voting process a V-pattern will indeed be formed in the diagonal direction in a single CIP in the Hough space if the object sought existed.

The first property gives the parallel efficiency to the AS-LIGHT algorithm while the second guarantees that the result is still correct.

4.3.2 Complexity Analysis for The AS-LIGHT Algorithm

The complexity analysis for the AS-LIGHT algorithm is as follows:

(a) It takes O(md) steps for the voting in one CIP and O(Nmd) steps for all N CIP's. Therefore, Step 2 (Vote Accumulation) has a time complexity of O(Nmd).

(b) Comparing the entries in the Hough space with their corresponding entries in the V-pattern and then putting the partial solutions together take O(L) time. Hence, the total running time for Step 3 (Pattern Matching) is O(NL).

To summarize, the total time complexity of the AS-LIGHT algorithm is O(N(md+L)) on a N×N image, provided that the LAP has N PE's. Compared with the sequential S-LIGHT algorithm, which has $O(N^2(m+L))$ running time, the AS-LIGHT algorithm can reduce the time complexity of the voting process by a factor of N/d and speed up the pattern matching process by a factor of N.

Normally, the number of rows in a VXY-table is limited. Take $d = 360 / \tau$, where τ is the increment of the gradient directions in the VXY-table. In our applications, τ is chosen between 5 and 20 degrees, so d is a constant and lies between 18 and 72. Given a normal image size of 512×512, i.e., N = 512, the speed-up for vote accumulation is approximately 7 to 28 times. One must bear in mind that the pattern matching of the AS-LIGHT algorithm has a speed-up factor N. So the overall speed-up is considerably higher than N/d for the entire recognition process.

4.4 The IAAS-LIGHT Algorithm for Array Processors with Indirect Addressing

In this section, it will be shown how the AS-LIGHT algorithm's vote accumulation time can be further reduced by a factor of d, if the LAP has the capability of indirect addressing. The new algorithm is thus named *Indirect Addressing Array processor S-LIGHT (IAAS-LIGHT)*.

In the AS-LIGHT algorithm, the central controller sequentially searches through d rows of the VXY-table. Each PE votes only when its edge point has the same gradient direction as the current ϕ_j. The indirect addressing capability will bring in two significant improvements: (a) each PE can use the ϕ of its edge point to independently look up the VXY-table; (b) each PE can vote to different addresses according to the independent table entry it found.

With the same assumptions made for the AS-LIGHT algorithm, the vote accumulation procedure for the IAAS-LIGHT algorithm can be described as follows:

IAAS-LIGHT Vote Accumulation Procedure:

For x = 1 to N do:

For i = 1 to m do:

Each PE_y (y = 1, ..., N) performs the following voting procedure at the same time:

if the x-th pixel is an edge point of gradient direction ϕ and the contents of VXY-table$(\phi)_i$ is $(x(\phi)_i, y(\phi)_i)$, then

$$x_c = x - x(\phi)_i,$$

$$z_c = y(\phi)_i,$$

$$A(x_c, y, z_c) = A(x_c, y, z_c) + 1 . \qquad (4.2)$$

endif

It takes O(m) steps to finish voting in one CIP, and O(Nm) steps for all N CIP's. Therefore, the IAAS-LIGHT voting procedure has the time complexity of O(Nm). Compared with the AS-LIGHT algorithm, the voting complexity is reduced by a factor of d. This is because with the indirect addressing ability all the PE's are able to read the different entries of the VXY-table at the same time and to perform voting to independent local memory locations at the same time.

The overall time complexity of the IAAS-LIGHT algorithm is O(N(m+L)) with N PE's on a N×N size image. Compared with the sequential S-LIGHT algorithm, which has $O(N^2(m+L))$ worst running time, the IAAS-LIGHT algorithm can reduce the running time complexity of the S-LIGHT algorithm by a factor of N with N PE's, and thus achieves a linear speed-up.[2]

---

2 The indirect addressing array processor is not available to us yet. It is encouraging to see that the next generation AIS parallel vision processor, the CENTIPEDE, as described by Wilson [17] will provide exactly this feature.

5. IMPLEMENTATIONS AND EXPERIMENTAL RESULTS

This section will demonstrate some experimental results for testing the sequential and parallel LIGHT algorithms presented in this paper.

5.1 Implementation of the S-LIGHT Algorithm

5.1.1 The Implementation

The sequential S-LIGHT algorithm is implemented in C under the UNIX environment on a Sun-3/280 workstation. Edges and their gradient directions are extracted by applying the 3×3 gradient operators. Afterwards, an edge thinning operator is applied to get a thin edge map. The construction of the VXY-table and the V-patterns on the Y axis is as described in Section 3. For dealing with the "key" test images, our pattern matching process adopts the following heuristic criteria:

(a) An entry in V-pattern matches to its corresponding entry in the Hough space if the two entries differ by less than 25 percent of the value in the V-pattern (tolerating small error on each entry).

(b) Contiguous unmatched entry pairs in a matched segment must be less than 5 in length and each unmatched pair must not differ more than 2 in number or 50 percent in value in the V-pattern (tolerating a certain amount of moderate error caused by breaks in a segment).

(c) A matched piece must exceed 15 percent of the length of the corresponding V-pattern (declaring a very small partial match is not meaningful).

The above criteria are obtained through experiments on several relatively good quality images. The actual numbers had to be adjusted for each particular set of image data. Such weak heuristics are adopted because of the lack of successful algorithms for inexact matching in complicated situations for real-world images.

5.1.2 Experimental Results

A key in a 256×256 image is used as our model object for testing (shown in Figure 5.1(a)). The extracted boundary for the key model is shown in Figure 5.1(b).

A No-occlusion Case

Figure 5.2(a) shows two keys without occlusion, one of the keys has the same shape as the model key. Figure 5.2(b) is the edge map. Figure 5.2(c) is the recognized and regenerated object which shows that the entire boundary of the key was recognized. Figure 5.3(a) graphically depicts the V-pattern for the key

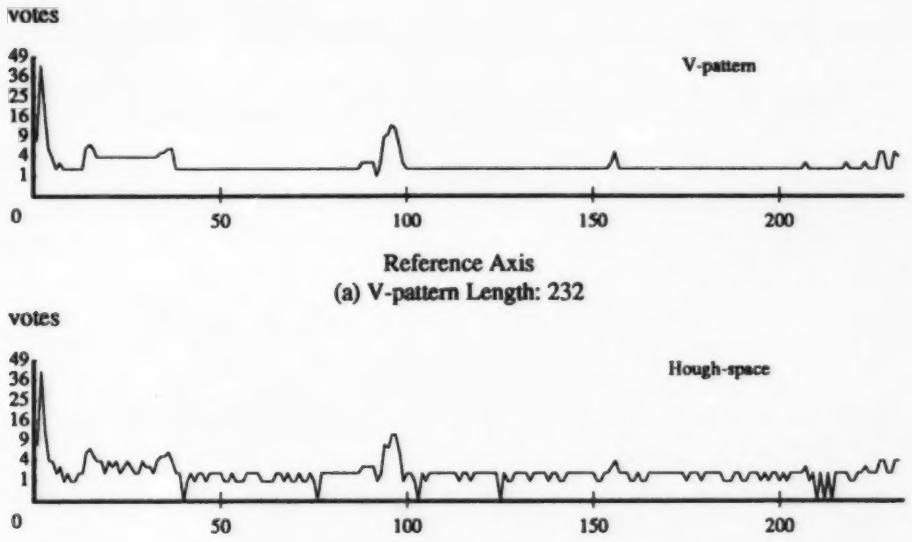
model (the horizontal axis in the graph corresponds to the vertical reference axis in the image). Figure 5.3(b) shows the voting result on the column at the reference point in the Hough space. For display purpose, the vertical axes in the figures have been scaled using the square root of votes. An undesirable effect is that the errors of low value entries appears to be graphically exaggerated. It is not hard to see that all the basic features were preserved in the accumulated column in the Hough space. Figure 5.4 shows the votes accumulated in the columns on the four neighboring points immediately next to the reference point. Apparently, the votes collected in those columns are not significant, and they will not hinder the recognition of the best matching at the reference point.

(a) (b)

Figure 5.1 Model Key and Its Boundary

(a) (b) (c)

Figure 5.2 No Occlusion Case



(a) V-pattern Length: 232

(b) The Column on the Reference Point in the Hough Space (Matched: 1-232)

Figure 5.3 V-pattern and the Matching Column in the 3D Hough Space

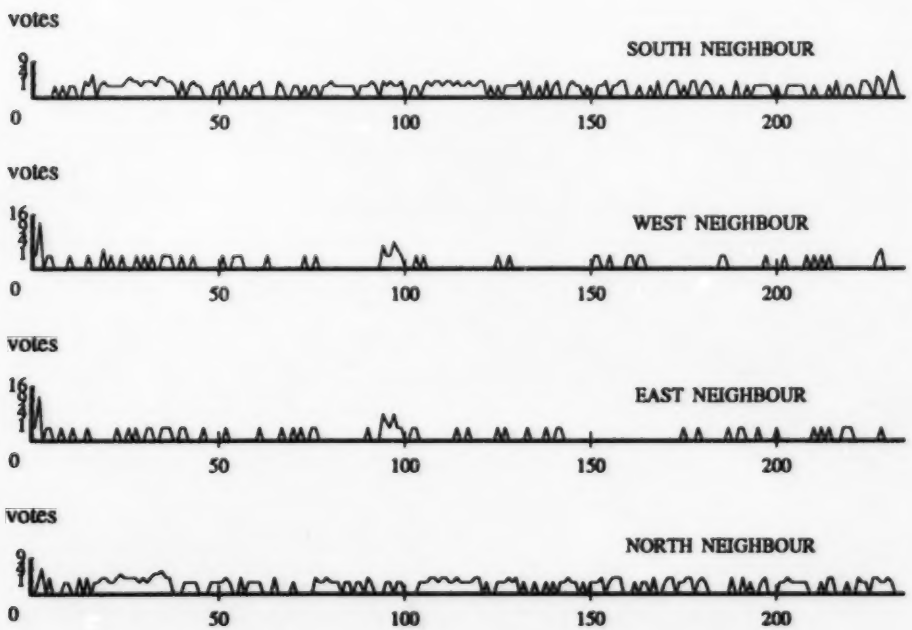
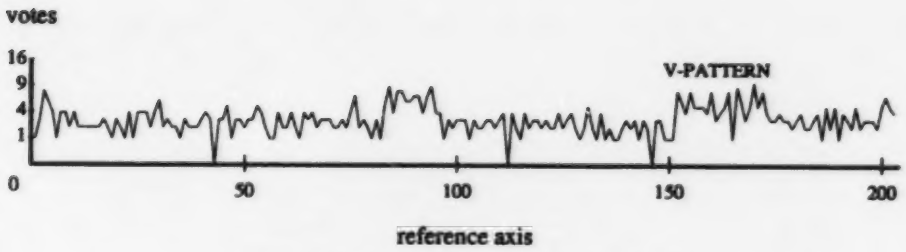


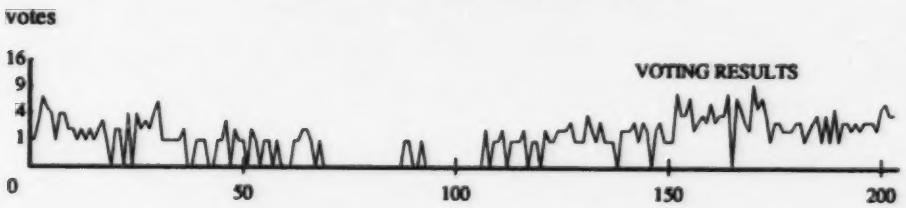
Figure 5.4 Voting Result in Columns of Four Immediate Neighbors

(a) (b) (c)

Figure 5.5 Occlusion Case



(a) Rotation: 140 degree. Pattern Length: 203



(b) Matched: 1-40, 125-203

Figure 5.6 Rotated V-patterns Used for Partial Matches

An Occluding Case

Figure 5.5 is very similar to Figure 5.2 except it presents an image of an occluded key. For recovering a maximal non-occluded part, an effective and shrewd method is to rotate both the model and the image so that the end points of the non-occluded boundary fall into the same row after the rotation. It is even possible to achieve maximal recovery of the occluded object by merging pattern matching results from multiple rotations [18]. In our implementation, the VXY-tables and V-patterns for all possible rotations are pre-constructed. Actually, there is no need to rotate the image. The same effect can be achieved by voting and pattern matching on a rotated reference axis, which incurs almost no additional overhead. Figures 5.6(a) shows the new V-pattern for the model key while the reference axis is rotated by 140 degree. Figure 5.6(b) is the voting result on the column at the reference point in the Hough space under the 140 degree rotation.

5.2 Implementation of the AS-LIGHT Algorithm on a LAP

5.2.1 The AIS-4000 Parallel Vision Processor

The AIS-4000 [16] is a commercially available SIMD 1×N array vision processor. A total of 512 bit-serial processors are arranged in a one-dimensional linear array with a direct communication link between adjacent processors. Each processor has 32k bit memory associated with it in which an entire row of an image can be stored. The rows of memory can also be used to store alternate images or the results of intermediate computations. The architecture is anisotropic in that the vertical dimension is massively parallel and the horizontal dimension is similar to a standard single-processor architecture with random memory access. This allows a virtual 2-dimensional SIMD programming model to be used, when necessary. A parallel-array controller is responsible for loading instructions into the array of processing elements. When processing of one column is completed, the parallel-memory addresses can be incremented to process the next and remaining columns in the image. This architecture results in low overhead for loading parallel-processor instructions and greatly simplifies the data-communication channels between both real processors and virtual processors (which are assigned to different columns in the image). Programming the AIS-4000 is accomplished through an extensive library of image-processing functions that are invoked through the C programming language. As a default, the library functions operate on an entire image using the virtual 2-dimensional SIMD model. However, operations can also be restricted to any subset of the image columns.

5.2.2 Experimental Results

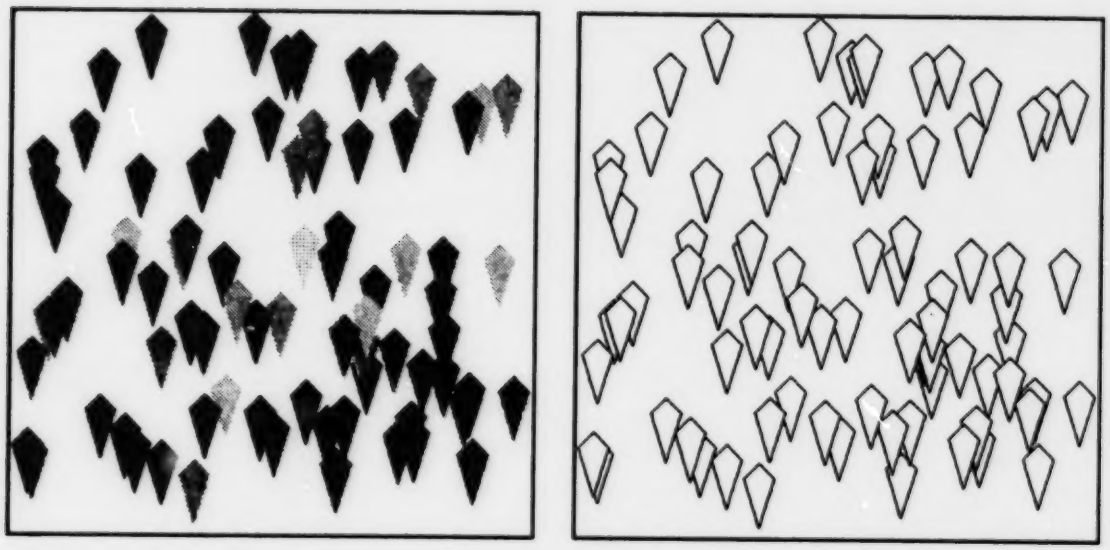
The AS-LIGHT algorithm has been implemented on the AIS-4000 system. A 512×512 test image was used to compare its performance against the sequential S-LIGHT. Figure 5.7 shows the test image and the

corresponding edge map. The image is generated by placing a kite-shaped pattern in random locations within the image frame. The task is to locate each of these kite patterns in the image. Occlusions present in the image would show the correctness of the algorithm in recognizing partially occluded objects. Figure 5.8 shows the kite model and its boundary map based on which a VXY-table is constructed to guide the voting process. In the experiment, all the kite patterns were correctly identified. The occluded patterns were also recognized with the visible segments matched successfully against the linear V-pattern of the model template. Figure 5.9 presents the result of the pattern matching.

The AS-LIGHT matches all the V-patterns accumulated in the 3D Hough space. For display purpose, only those V-patterns which have more than 50% matched entries are shown. The matched V-patterns are drawn as diagonals standing on their respective reference points (the lowest tip of the kite pattern). Contiguous matched entries signifying a visible portion of an occluded pattern are represented as a dark segment on the diagonal.

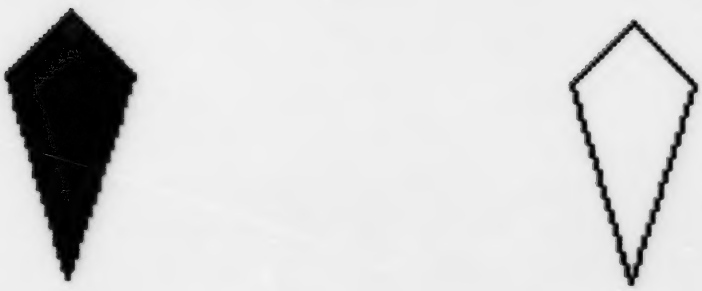
A straight-forward implementation of the S-LIGHT and AS-LIGHT requires a space of $O(N^2L)$ for storing the 3D Hough space, where $N \times N$ is the size of the image and L is the length of V-pattern. Such a space requirement is not practical for implementation. The AIS-4000 has a local memory of only 32k bits at each PE. It cannot accommodate a Hough space of large size. However, in many cases, it is not necessary to have the entire 3D Hough space resident in memory at the same time. In our method, the V-pattern matching proceeds while the votes in the Hough space are matched against the model V-pattern one entry at a time. The VXY-table can also be sorted in such a way that the voting to different horizontal slices of the Hough space can be ordered. This enables the Hough space to be processed one slice at a time. In our implementation, the space requirement of the AS-LIGHT program is reduced from $O(N^2L)$ to $O(N^2)$, which makes it possible to run under the memory constraint of the AIS-4000.

The execution time of the AS-LIGHT compares very favorably to the sequential S-LIGHT. In the experiments, the parallel voting took about 0.47 second and the subsequent V-pattern matching finished in 0.46 second. For comparison, we have also implemented a sequential S-LIGHT algorithm similar to the AS-LIGHT on a Sun-3/280 and a SUN SPARCstation-1. On the Sun-3/280, the S-LIGHT program took 60 seconds to perform the sequential voting and about 200 seconds for the V-pattern matching. On the SUN SPARCstation-1, the program ran slightly faster, 40 seconds for the voting and 120 seconds for the matching. The CPU timing results are tabulated in Table 1. The AS-LIGHT running on the AIS-4000 did show a substantial speed-up in execution time. A factor of two orders of magnitude was achieved.



(a) (b)

Figure 5.7 A 512×512 test image of kite patterns and the corresponding edge map



(a) (b)

Figure 5.8 The kite model and its boundary map 64×32

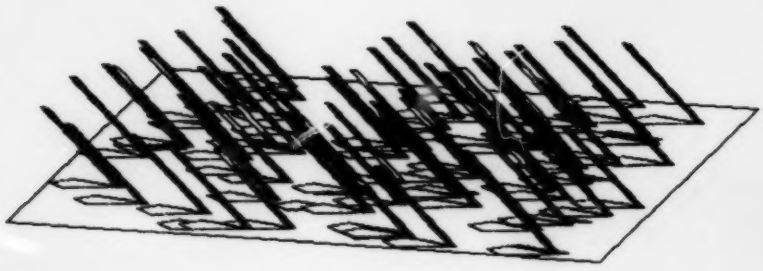


Figure 5.9 The 3D Hough space with the diagonal V-patterns standing on the reference points of the matched patterns. Matched V-pattern entries are marked as dark and occluded parts are left blank.

Table 1 CPU Timing of the AS-LIGHT on AIS-4000, and the sequential S-LIGHT on SPARCstation 1 and Sun-3/280, respectively

	AIS-4000	SPARCstation 1	Sun-3/280
Vote Accumulation	0.47 sec.	40 sec.	60 sec.
Pattern Matching	0.46 sec.	120 sec.	200 sec.
Total	0.93 sec.	160 sec.	260 sec.

6. CONCLUSION

In this paper, we have shown a useful extension to Ballard's Generalized Hough Transform (GHT) [3]. The GHT is a very powerful tool in image analysis. However, the GHT has some limitations. First, it lacks robustness. It may erroneously declare matches to the shapes similar to the model. It has difficulty in dealing with partially occluded objects, because the only criterion available is the peak value. Second, the GHT is not very suitable for parallel processing, since all the points on the boundary of the shape will cast votes to the reference point, and hence cause severe contention.

The essence of our Linear Generalized Hough Transform (LIGHT) is to use a linear pattern to replace the single reference point used in the GHT. The new technique has advantages over the GHT in all above aspects. The linear V-pattern in the LIGHT increases the robustness of the GHT by requiring that the votes accumulated on the reference point obey a certain linear distribution before declaring a match. It also provides a systematic approach to deal with occlusion. Partial matches of V-patterns to the Hough space show the existence of partial objects. Moreover, the extended voting space also alleviates the contention problem in parallel processing.

It has been shown that a simple P-LIGHT algorithm can be used for simple cases (e.g., when there is only a single object in the image), whereas a more robust S-LIGHT algorithm is needed to maximize the recovery of the visible boundaries of the occluded objects. The analysis of the time complexity consists of two parts: vote accumulation and pattern matching. For vote accumulation, the LIGHT algorithms (P-LIGHT and S-LIGHT) have the same time complexity $O(N^2m)$ as the GHT, where m is the width of the VX- or VXY-table constructed for the model, although the P-LIGHT algorithm usually has a smaller m and is faster in vote accumulation. For pattern matching, the S-LIGHT algorithm has a time complexity $O(N^2L)$, which is L times worse than the GHT (L is the length of the object along the Y-axis). It is because that the S-LIGHT algorithm must match linear patterns while the GHT looks for single peak points. The S-LIGHT algorithm employs a 3D Hough space, therefore, the space complexity of its straight-forward version is a factor of L worse than both the GHT and P-LIGHT algorithms. However, with the pattern matching method proposed in this paper, the 3D Hough space can be processed a slice at a time. In our implementation of the S-LIGHT algorithm, the space requirement has been reduced from $O(N^2L)$ to $O(N^2)$.

It has also been shown that the S-LIGHT algorithm can be implemented on two types of Linear Array processors (LAP's). The first parallel version of the S-LIGHT algorithm, AS-LIGHT, is based on a very basic linear array processor. It is shown that the speed-up for the vote accumulation is N/d with N PE's, where d is a pre-determined constant representing the number of rows in the VXY-table. The second parallel version, IAAP-LIGHT, which will run on a LAP with independent addressing ability, is also

proposed. The IAAP-LIGHT could achieve linear speed-up for the vote accumulation. Both parallel algorithms can achieve linear speed up for the pattern matching in a LAP.

Experimental results are presented to illustrate the recognition of partially occluded objects from both real-images and synthetic images. Array processors with indirect addressing ability is under construction (e.g. the CENTIPEDE [17]) and not available as yet. Our result for parallel implementation is obtained from an SIMD one-dimensional array processor (the AIS-4000 Vision Processor with 512 processing elements). It has been shown that substantial speed-up is achieved. In our experiments, the parallel AS-LIGHT program took 0.93 second to analyze a 512×512 image using a 64×32 model template. A similar sequential S-LIGHT program took 160 seconds on a Sun SPARCstation-1 and 260 seconds on a Sun-3/280. A speed-up of two orders of magnitude was achieved.

Clearly, the search for V-patterns in the Hough space should tolerate certain errors by using inexact matching scheme for real world images. More research is needed in inexact matching algorithms before a more robust LIGHT technique for object recognition can be developed.

7. ACKNOWLEDGMENTS

This work is supported in part by the Canadian National Science and Engineering Research Council under Grants A-36726 and EQ-42445.

8. REFERENCES

[1] Adiv, G. Recovering motion parameters in scenes containing multiple moving objects. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Washington, pp. 399-400, 1983.

[2] Aho, A.V., Hopcroft, J.E. and Ullman, J.D., The design and analysis of computer algorithms. *Addison-Wesley*, 1974.

[3] Ballard, D.H. Generalized the Hough transform to detect arbitrary shapes. *Pattern Recognition*, vol. 13, pp. 111-122, 1981.

[4] Ballard, D.H. and Sabbah, D. Viewer independent shape recognition. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 5, pp. 653-660, 1983.

[5] Batcher, K.E. Design of a massively parallel processor. *IEEE Transactions on Computers*, vol. 29, No. 9, pp. 826-840, 1980.

[6] Cypher, R.E., Sanz, J.L.C. and Snyder, L. The Hough Transform has O(N) complexity on SIMD NxN mesh array architectures. *Proceedings of IEEE Workshop on Computer Architectures For Pattern Analysis and Machine Intelligence*, Seattle, pp. 115-121, 1987.

[7] Davis, L.S. Hierarchical generalized Hough transforms and line segment based generalized Hough transforms. *Pattern Recognition*, vol. 15, pp. 277-285, 1982.

[8] Davies, E.R. Corner detection using the generalized Hough transform. *IEE 2nd International Conference on Image Processing and its Applications*, London, pp. 175-179, 1986.

[9] Grimson, W.E.L. and Huttenlocher, D.P. On the sensitivity of the Hough transform for object recognition. *Proceedings of the 2nd International Conference on Computer Vision*, pp. 700-706, 1988.

[10] Illingworth, J. and Kittler, J. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 87-116, 1988.

[11] Kasif, S., Kitchen, L., and Rosenfeld, A. A Hough transform technique for subgraph isomorphism. *Pattern Recognition Letter*, vol. 2, pp. 83-88, 1983.

[12] Little, J.J., Blelloch, G., and Cass, T. Parallel algorithms for computer vision on the connection machine. *Proceedings of the 1st IEEE International Conference on Computer Vision*, London, pp. 587-591, 1987.

[13] Merlin, P.M. and Farber, D.J. A parallel mechanism for detecting curves in pictures. *IEEE Transactions on Computers*, vol. 24, pp. 96-98, 1975.

[14] Olson, T.J., Bukys, L., and Brown, C.M. Low level image analysis on an MIMD architecture. *Proceedings of the 1st IEEE International Conference on Computer Vision*, London, pp. 468-475, 1987.

[15] Reeves, A.P. Survey: parallel computer architectures for image processing. *Computer Vision, Graphics, and Image Processing*, vol. 25, pp. 68-88, 1984.

[16] Schmitt, L.A. and Wilson, S.S. The AIS-5000 Parallel Processor. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol. 10, No. 3, pp. 320-329, May 1988.

[17] Wilson, S.S. One dimensional SIMD Architectures--The AIS-5000. *Multicomputer Vision*, Academic Press, London, pp. 131-149, 1988.

[18] Yao, B and Li, Z.N., Detecting partially occluded objects using an extended generalized Hough transform. *Proceedings IEEE International Conference on Image Processing*, pp. 573-577, 1989.

Centre for Systems Science

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

• CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.